Network Working Group
Request for Comments: 891

D.L. Mills
December 1983

DCN Local-Network Protocols

This RFC is a description of the protocol used in the DCN local networks to maintain connectivity, routing, and timekeeping information. These procedures may be of interest to designers and implementers of other networks.

1. Introduction

This document describes the local-net architecture and protocols of the Distributed Computer Network (DCN), a family of local nets based on Internet technology and an implementation of PDP11-based software called the Fuzzball. DCN local nets have been in operation for about three years and now include clones in the USA, UK, Norway and West Germany. They typically include a number of PDP11 or LSI-11 Fuzzballs, one of which is elected a gateway, and often include other Internet-compatible hosts as well.

The DCN local-net protocols are intended to provide connectivity, routing and timekeeping functions for a set of randomly connected personal computers and service hosts. The design philosophy guiding the Fuzzball implementation is to incorporate complete functionality in every host, which can serve as a packet switch, gateway and service host all at the same time. When a set of Fuzzballs are connected together using a haphazard collection of serial, parallel and contention-bus interfaces, they organize themselves into a network with routing based on minimum delay.

The purpose of this document is to describe the local-net protocols used by the DCN to maintain connectivity, routing and timekeeping functions. The document is an extensive revision and expansion of Section 4.2 of [1] and is divided into two parts, the first of which is an informal description of the architecture, together with explanatory remarks. The second part consists of a semi-formal specification of the entities and protocols used to determine connectivity, establish routing and maintain clock synchronization and is designed to aid in the implementation of cohort systems. The link-level coding is described in the appendix.

2. Narrative Description

The DCN architecture is designed for local nets of up to 256 hosts and gateways using the Internet Protocol (IP) and client protocols. It provides adaptive routing and clock synchronization functions in an arbitrary topology including point-to-point links and multipoint bus systems. It is intended for use in connecting personal computers to each other and to service machines, gateways and other hosts of the Internet community. However, it is not intended for use in large, complex networks and does not support the sophisticated routing and control algorithms of, for example, the ARPANET.

A brief description of the process and addressing structure used in the DCN may be useful in the following. A DCN physical host is a PDP11-compatible processor which supports a number of cooperating sequential processes, each of

which is given a unique 8-bit identifier called its port ID. Every DCN physical host contains one or more internet processes, each of which supports a virtual host given a unique 8-bit identifier called its host ID.

Each virtual host can support multiple internet protocols, connections and, in addition, a virtual clock. Each physical host contains a physical clock which can operate at an arbitrary rate and, in addition, a 32-bit logical clock which operates at 1000 Hz and is assumed to be reset each day at 0000 hours UT. Not all physical hosts implement the full 32-bit precision; however, in such cases the resolution of the logical clock may be somewhat less.

There is a one-to-one correspondence between Internet addresses and host IDs. The host ID is formed from a specified octet of the Internet address to which is added a specified offset. The octet number and offset are selected at configuration time and must be the same for all DCN hosts sharing the local net. For class-B and class-C nets normally the fourth octet is used in this way for routing within the local net. In the case of class-B nets, the third octet is considered part of the net number by DCN hosts; therefore, this octet can be used for routing between DCN local nets. For class-A nets normally the third octet (ARPANET logical-host field) is used for routing where necessary.

Each DCN physical host is identified by a host ID for the purpose of detecting loops in routing updates, which establish the minimum-delay paths between the virtual hosts. By convention, the physical host ID is assigned as the host ID of one of its virtual hosts. A link to a neigbor net is associated with a special virtual host, called a gateway, which is assigned a unique host ID.

The links connecting the various physical hosts together and to foreign nets can be distributed in arbitrary ways, so long as the net remains fully connected. If full connectivity is lost, due to a link or host fault, the virtual hosts in each of the surviving segments can continue to operate with each other and, once connectivity is restored, with all of them.

Datagram routing is determined entirely by internet address - there is no local leader as in the ARPANET. Each physical host contains two tables, the Host Table, which is used to determine the outgoing link to each other local-net host, and the Net Table, which is used to determine the outgoing host (gateway) to each other net. The Host Table contains estimates of roundtrip delay and logical-clock offset for all virtual hosts in the net and is indexed by host ID. For the purpose of computing these estimates the delay and offset of each virtual host relative to the physical host in which it resides is assumed zero. The single exception to this is a special virtual host associated with an NBS radio time-code receiver, where the offset is computed relative to the broadcast time.

The Net Table contains an entry for every neighbor net that may be connected to the local net and, in addition, certain other nets that are not

neighbors. Each entry contains the net number, as well as the host ID of the local-net gateway to that net. The routing function simply looks up the net number in the Net Table, finds the host ID of the gateway and retrieves the port ID of the net-output process from the Host Table. Other information is included in the Host Table and Net Table as discribed below.

The delay and offset estimates are updated by HELLO messages exchanged on the links connecting physical-host neighbors. The HELLO messages are exchanged frequently, but not so as to materially degrade the throughput of the link for ordinary data messages. A HELLO message contains a copy of the delay and offset information from the Host Table of the sender, as well as information to compute the roundtrip delay and logical-clock offset of the receiver relative to the sender.

The routing algorithm is similar to that (formerly) used in the ARPANET and other places in that the roundtrip (biased) delay estimate calculated to a neighbor is added to each of the delay estimates given in its HELLO message and compared with the corresponding delay estimates in the Host Table. If a delay computed in this way is less than the delay already in the Host Table, the routing to the corresponding virtual host is changed accordingly. The detailed operation of this algorithm, which includes provisions for host up-down logic and loop suppression, is summarized in a later section.

DCN local nets are self-configuring for all hosts and neighbor nets; that is, the routing algorithms will find minimum-delay paths between all hosts and gateways to neighbor nets. In addition, timekeeping information can be exchanged using special HELLO messages between neighboring DCN local nets. For routing beyond neighbor nets additional entries can be configured in the Net Tables as required. In addition, a special entry can be configured in the Net Tables which specifies the host ID of the gateway to all nets not explictly included in the table.

For routing via the ARPANET and its reachable nets a selected local-net host is equipped with an IMP interface and configured with a GGP/EGP Gateway process. This process maintains the Net Table of the local host, including ARPANET leaders, dynamically as part of the GGP/EGP protocol interactions with other ARPANET gateways. GGP/EGP protocol interactions are possibly with non-ARPANET gateways as well.

The portable virtual-host structure used in the DCN encourages a rather loose interpretation of addressing. In order to minimize confusion in the following, the term "host ID" will be applied only to virtual hosts, while "host number" will be applied to the physical host, called generically the DCN host.

2.1. Net and Host Tables

There are two tables in every DCN host which control routing of Internet Protocol (IP) datagrams: the Net Table and the Host Table. The Net Table is used to determine the host ID of the gateway on the route to a foreign net,

while the Host Table is used to determine the link, with respect to the DCN host, on the route to a virtual host. The Host Table is maintained dynamically using updates generated by periodic HELLO messages. The Net Table is fixed at configuration time for all DCN hosts except those that support a GGP/EGP Gateway process. In these cases the Net Table is updated as part of the gateway operations. In addition, entries in either table can be changed by operator commands.

The Net Table format is shown in Figure 1.

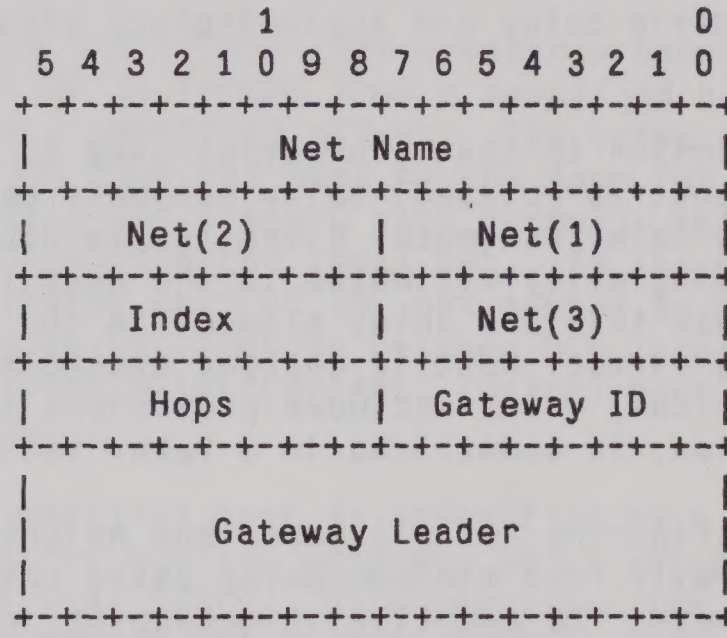


Figure 1. Net Table Entry

The "Net Name" field defines a short (RAD50) name for the net, while the "Net" fields define the class A/B/C net number. The "Gateway ID" field contains the host ID of the first gateway to the net and the "Hops" field the number of hops to it. The remaining fields are used only by the GGP/EGP Gateway process and include the "Index" field, which contains an index into the routing matrix. and the "Gateway Leader" field, which contains the (byte-swapped) local-net leader for the gateway on a neighbor net.

The Net Table contains an indefinite number of entries and is terminated by a special entry with all "Net" fields set to zero. If the "Hops" field of this entry is less than 255, the "Gateway ID" field of this entry is used for all nets not in the table. If the "Hops" field is 255 all nets not explicitly mentioned in the table appear unreachable.

The Host Table format is shown in Figure 2.

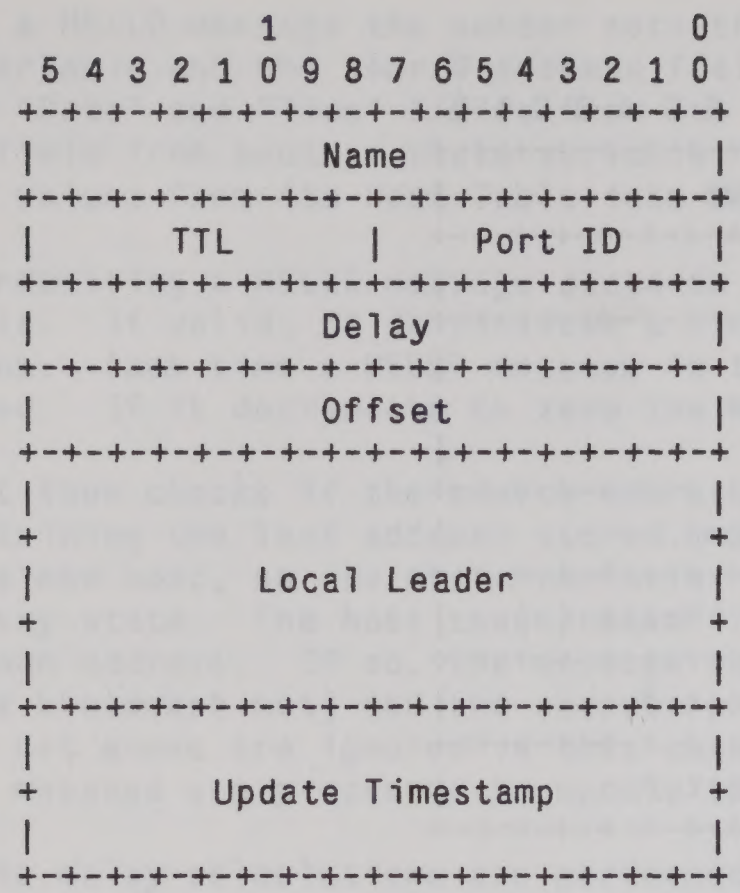


Figure 2. Host Table Entry

The ordinal position of each Host Table entry corresponds to its host ID. The "Name" field contains a short (RAD50) name for convenient reference. The "Port ID" field contains the port ID of the net-output process on the shortest path to this virtual host and the "Delay" field contains the measured roundtrip delay to it. The "Offset" field contains the difference between the logical clock of this host and the logical clock of the local host. The "Local Leader" field contains information used to construct the local leader of the outgoing packet, for those nets that require it. The "Update Timestamp" field contains the logical clock value when the entry was last updated and the "TTL" field the time (in seconds) remaining until the virtual host is declared down.

All fields except the "Name" field are filled in as part of the routing update process, which is initiated upon arrival of a HELLO message from a neighboring DCN host. This message takes the form of an IP datagram carrying the reserved protocol number 63 and a data field as shown in Figure 3.

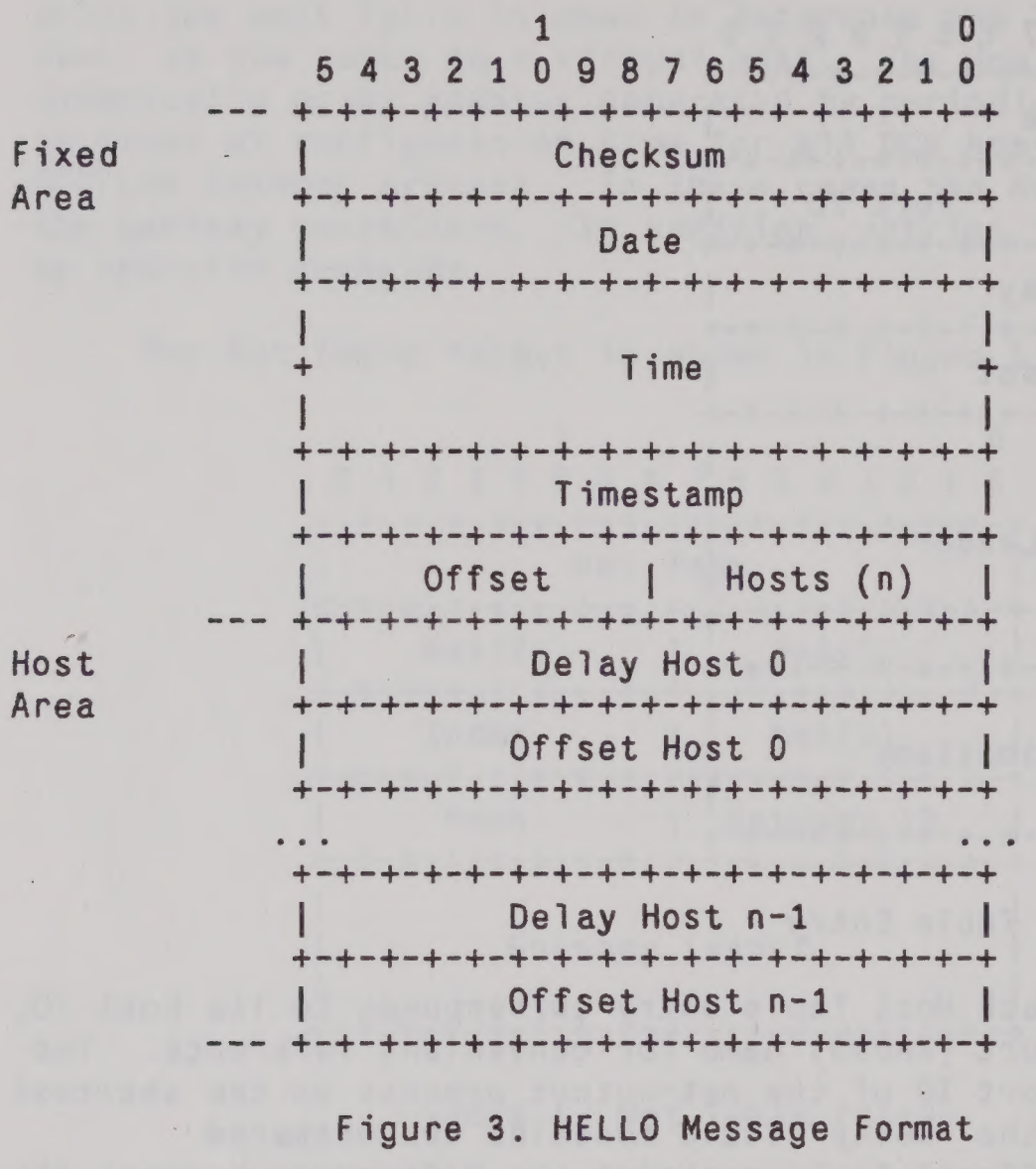


Figure 3. HELLO Message Format

There are two HELLO message formats, depending on the length of the message. One format, sent by a DCN host to another host on the same local net, includes both the fixed and host areas shown above. The second format, sent in all other cases, includes only the fixed area.

Note that all word fields shown are byte-swapped with respect to the ordinary PDP11 representation. The "Checksum" field contains a checksum covering the fields indicated. The "Date" and "Time" fields are filled in with the local date and time of origination. The "Timestamp" field is used in the computation of the roundtrip delay (see below). The "Offset" field contains the offset of the block af Internet addresses used by the local net. The "Delay Host n" and "Offset Host n" fields represent a copy of the corresponding entries of the Host Table as they exist at the time of origination. The "Hosts (n)" field contains the number of entries in this table.

2.2. Roundtrip Delay Calculations

Periodically, each DCN physical host sends a HELLO message to its neighbor on each of the communication links common to both of them. For each of these links the sender keeps a set of state variables, including a copy of the source-address field of the last HELLO message received. When

constructing a HELLO message the sender sets the destination-address field to this state variable and the source-address field to its own address. It then fills in the "Date" and "Time" fields from its logical clock and the "Timestamp" field from another state variable. It finally copies the "Delay" and "Offset" values from its Host Table into the message.

A host receiving a HELLO message discards it if the format is bad or the checksum fails. If valid, it initializes a link state variable to show that the link is up. Each time a HELLO message is transmitted this state variable is decremented. If it decrements to zero the link is declared down.

The host then checks if the source-address field matches the state variable containing the last address stored. If not, the link has been switched to a new host, so the state variables are flushed and the link forced into a recovery state. The host then checks if the destination-address field matches its own address. If so, the message has been looped (legal only in the case of a broadcast net) and the roundtrip delay information is corrected. The host and net areas are ignored in this case. If not, the host and net areas of the message are processed to update the Host and Net Tables.

Roundtrip delay calculations are performed in the following way. The link input/output processes assigned each link maintain an internal state variable which is updated as each HELLO message is received and transmitted. When a HELLO message is received this variable takes the value of the "Time" field minus the current time-of-day. When the next HELLO message is transmitted, the value assigned the "Timestamp" field is computed as the low-order 16-bits of this variable plus the current time-of-day. The value of this variable is forced to zero if either the link is down of the system logical clock has been reset since the last HELLO message was received.

If a HELLO message is received with zero "Timestamp" field, no processing other than filling in the internal state variable. Otherwise, the roundtrip delay is computed as the low-order 16-bits of the current time-of-day minus the value of this field. In order to assure the highest accuracy, the calculation is performed only if the length of the last transmitted HELLO message (in octets) matches the length of the received HELLO message.

The above technique renders the calculation independent of the clock offsets and intervals between HELLO messages at either host, protects against errors that might occur due to lost HELLO messages and works even when a neighbor host simply forwards the HELLO message back to the originator without modifying it. The latter behavior, typical of ARPANET IMPs and gateways, as well as broadcast nets, relys on the loop-detection mechanism so that correct calculations can be made and, furthermore, that spurious host updates can be avoided.

2.3. Host Updates

When a HELLO message arrives which results in a valid roundtrip delay calculation, a host update process is performed. This consists of adding the

roundtrip delay to each of the "Delay Host n" entries in the HELLO message in turn and comparing each of these calculated delays to the "Host Delay" field of the corresponding Host Table entry. Each entry is then updated according to the following rules:

1. If the link connects to another DCN host on the same net and the port ID (PID) of the link output process matches the "Port ID" field of the entry, then update the entry.

2. If the link connects to another DCN host on the same net, the PID of the link output process does not match the "Port ID" field and the calculated delay is less than the "Host Delay" field by at least a specified switching threshold (currently 100 milliseconds), then update the entry.

3. If the link connects to a foreign net and is assigned a host ID corresponding to the entry, then update the entry. In this case only, use as the calculated delay the roundtrip delay.

4. If none of the above conditions are met, or if the virtual host has been declared down and the "TTL" field contains a nonzero value, then no update is performed.

The update process consists of replacing the "Delay" field with the calculated delay, the "Port ID" field with the PID of the link output process, the "Update Timestamp" field with the current time of day and the "TTL" field by a specified value (currently 120) in seconds. If the calculated delay exceeds a specified maximum interval (currently 30 seconds), the virtual host is declared down by setting the corresponding "Delay" field to the maximum and the remaining fields as before. For the purposes of delay calculations values less than a specified minimum (currently 100 milliseconds) are rounded up to that minimum.

The "Offset" field is also replaced during the update process. When the HELLO message arrives, The value of the current logical clock is subtracted from the "Time" field and the difference added to one-half the roundtrip delay. The resulting sum, which represents the offset of the local clock to the clock of the sender, is added to the corresponding "Offset" field of the HELLO message and the sum replaces the "Offset" field of the Host Table. Thus, the "Offset" field in the Host Table for a particular virtual host is replaced only if that host is up and on the minimum-delay path to the DCN host.

The purpose of the switching threshold in (2) above and the minimum delay specification in the update process is to avoid unnecessary switching between links and transient loops which can occur due to normal variations in propagation delays. The purpose of the "TTL" field test in (4) above is to insure consistantcy by purging all paths to a virtual host when that virtual host goes down.

In addition to the updates performed as HELLO messages arrive, each

virtual host in a DCN host also performs a periodic update of its own Host Table entry. The update procedure is identical to the above, except that the calculated delay and offset are taken as zero. At least one of the virtual hosts in a DCN host must have the same host ID as the host number assigned the DCN host itself and all must be assigned the same net number. Other than these, there are no restrictions on the number or addresses of internet processes resident in a single DCN host.

It should be appreciated that virtual hosts are truly portable and can migrate about the net, should such a requirement arise. The host update protocols described here insure that the routing procedures always converge to the minimum-delay paths via operational links and DCN hosts. In the case of broadcast nets such as Ethernets, the procedures are modified slightly as described below. In this case the HELLO messages are used to determine routing from the various Ethernet hosts to destinations off the cable, as well as to provide time synchronization.

2.4. Timeouts

The "TTL" field in every Host Table entry is decremented once a second in normal operation. Thus, if following a host update another update is not received within an interval corresponding to the value initialized in that field, it decrements to zero, at which point the virtual host is declared down and the Host Table entry set as described above. The 120-second interval used currently provides for at least four HELLO messages to be generated by every neighbor on every link during that interval, since the maximum delay between HELLO messages is 30 seconds on the lowest-speed link (1200 bps). Thus, if no HELLO messages are lost, the maximum number of links between any virtual host and any other is four.

The "TTL" field is initialized at 120 seconds when an update occurs and when the virtual host is declared down. During the interval this field decrements to zero immediately after being declared down, updates are ignored. This provides a decent interval for the bad news to propagate throughout the net and for the Host Tables in all DCN hosts to reflect the fact. Thus, the formation of routing loops is prevented.

The IP datagram forwarding procedures call for decrementing the "time-to-live" field in the IP header once per second or at each point where it is forwarded, whichever comes first. The value used currently for this purpose is 30, so that an IP datagram can live in the net no longer than that number of seconds. This is thus the maximum delay allowed on any path between two virtual hosts. If this maximum delay is exceeded in calculating the roundtrip delay for a Host Table entry, the corresponding virtual host will be declared down.

The interval between HELLO messages on any link depends on the data rate supported by the link. As a general rule, this interval is set at 16 times the expected roundtrip time for the longest packet to be sent on that link. For 1200-bps asynchronous transmission and packet lengths to 256 octets, this

corresponds to a maximum HELLO message interval of about 30 seconds.

Although the roundtrip delay calculation, upon which the routing process depends, is relatively insensitive to net traffic and congestion, stochastic variations in the calculated values ordinarily occur due to coding (bit or character stuffing) and medium perturbations. In order to suppress loops and needless path changes a minimum switching threshold is incorporated into the routing mechanism (see above). The interval used for this threshold, as well as for the minimum delay on any path, is 100 milliseconds.

3. Formal Specification

The following sections provide a formal framework which describe the DCN HELLO protocol. This protocol is run between neighboring DCN hosts that share a common point-to-point link and provides automatic connectivity determination, routing and timekeeping functions.

The descriptions to follow are organized as follows: First a summary of data structures describes the global variables and packet formats. Then three processes which implement the protocol are described: the CLOCK, HELLO and HOST processes. The description of these processes is organized into sections that describe (1) the local variables used by that process, (2) the parameters and constants and (3) the events that initiate processing together with the procedures they evoke. In the case of variables a distinction is made between state variables, which retain their contents between procedure calls, and temporaries, which have a lifetime extending only while the process is running. Except as noted below, the initial contents of state variables are unimportant.

3.1. Data Structures

3.1.1. Global Variables

ADDRESS
: This is a 32-bit bit-string temporary variable used to contain an Internet address.

CLOCK-HID
: This is an eight-bit integer state variable used to contain the host ID of the local-net host to be used as the master clock. It is initialized to the appropriate value depending upon the net configuration.

DATE
: This is a 16-bit bit-string state variable used to contain the date in RT-11 format. Bits 0-4 contain the year, with zero corresponding to 1972, bits 5-9 contain the day of the month and bits 10-14 contain the month, starting with one for January.

DATE-VALID
: This is a one-bit state variable used to indicate whether the local date

and time are synchronized with the master clock. A value of one indicates the local clock is not synchronized with the master clock. This variable is set to one initially and when the local time-of-day rolls over past midnight. It is set to zero each time a valid date and time update has been received from the master clock.

DELAY
This is a 16-bit integer temporary variable which represents the roundtrip delay in milliseconds to a host.

HID
This is an eight-bit integer temporary variable containing the host ID of some host on the local net.

There is a one-to-one correspondence between the Internet addresses of local hosts and their HIDs. The mapping between them is selected on the basis of the octet number of the Internet address. For DCN hosts it is the fourth octet, while for hosts directly connected to a class-A ARPANET IMP or gateway, it is the third octet (logical-host field). The contents of this octet are to be added to ADDRESS-OFFSET to form the HID associated with the address.

HOLD
This is an eight-bit counter state variable indicating whether timestamps are valid or not. While HOLD is nonzero, timestamps should be considered invalid. When set to some nonzero value, the counter decrements to zero at a 1-Hz rate. Its initial value is zero.

HOST-TABLE
This is a table of NHOSTS entries indexed by host ID (HID). There is one entry for each host in the local net. Each entry has the following format:

HOST-TABLE.DELAY
This is a 16-bit field containing the computed roundtrip delay in milliseconds to host HID.

HOST-TABLE.OFFSET
This is a 16-bit field containing the computed signed offset in milliseconds which must be added to the local apparent clock to agree with the apparent clock of host HID.

HOST-TABLE.PID
This is an eight-bit field containing the PID of the net-output process selected by the routing algorithm to forward packets to host HID.

HOST-TABLE.TTL
This is an eight-bit field used as a time-to-live indicator. It is decremented by the HOST process once each second and initialized to a

chosen value when a HELLO message is received.

The table is initialized with the HOST-TABLE.DELAY field set to MAXDELAY for all entries. The contents of the other fields are unimportant.

LOCAL-ADDRESS
This is a 32-bit bit-string state variable used to contain the local host Internet address.

NET-TABLE
This is a table of NNETS entries with the following format:

NET-TABLE.HID
This is an eight-bit field containing the host ID of the pseudo-process to forward packets to the NET-TABLE.NET net.

NET-TABLE.NET
This is a 24-bit field containing an Internet class-A (eight bits), class-B (16 bits) or class-C (24 bits) net number. Note that the actual field width for class-B net numbers is 24 bits in order to provide a subnet capability, in which the high-order eight bits of the 16-bit host address is interpreted as the subnet number.

The table is constructed at configuration time and must include an entry for every net that is a potential neighbor. A neighbor net is defined as a net containing a host that can be directly connected to a host on the local net. The entry for such a net is initialized with NET-TABLE.NET set to the neighbor net number and NET-TABLE.HID set to an arbitrary vitual-host ID not assigned any other local-net virtual host.

The remaining entries in NET-TABLE are initialized at initial-boot time with the NET-TABLE.NET fields set to zero and the NET-TABLE.HID fields set to a configuration-selected host ID to be used to forward packets to all nets other than neighbor nets. In the case where a gateway module is included in the local host configuration, the GGP and/or EGP protocols will be used to maintian these entries; while, in the case where no gateway module is included, only one such entry is required.

OFFSET
This is a 16-bit signed integer temporary variable which represents the offset in milliseconds to be added to the apparent clock time to yield the apparent clock time of the neighbor host.

3.1.2. Parameters

ADDRESS-OFFSET
This is an integer which represents the value of the Internet address field corresponding to the first host in HOST-TABLE.

NHOSTS

This is an integer which defines the number of entries in HOST-TABLE.

NNETS
This is an integer which defines the number of entries in MET-TABLE.

3.1.3. HELLO Packet Fields

PKT.ADDRESS-OFFSET
This eight-bit is copied from ADDRESS-OFFSET by the sender.

PKT.DATESTAMP
Bits 0-14 of this 16-bit field are copied from DATE by the sender, while bit 15 is copied from DATE-VALID.

PKT.DATE-VALID
This one-bit field is bit 15 of PKT.DATESTAMP.

PKT.DESTINATION
This 32-bit field is part of the IP header. It is copied from HLO.NEIGHBOR-ADDRESS by the sender.

PKT.HOST-TABLE
This is a table of PKT.NHOSTS entries, each entry of which consists of two fields. The entries are indexed by host ID and have the following format:

PKT.HOST-TABLE.DELAY
This 16-bit field is copied from the corresponding HOST-TABLE.DELAY field by the sender.

PKT.HOST-TABLE.OFFSET
This 16-bit field is copied from the corresponding HOST-TABLE.OFFSET field by the sender.

PKT.LENGTH
This 16-bit field is part of the IP header. It is set by the sender to the number of octets in the packet.

PKT.NHOSTS
This eight-bit field is copied from NHOST by the sender.

PKT.SOURCE
This 16-bit field is part of the IP header. It is copied from LOCAL-ADDRESS by the sender.

PKT.TIMESTAMP
This 32-bit field contains the apparent time the packet was transmitted in milliseconds past midnight UT.

PKT.TSP
This 16-bit field contains a variable used in roundtrip delay

calculations.

3.2 CLOCK Process (CLK)

The timekeeping system maintains three clocks: (1) the physical clock, which is determined by a hardware oscillator/counter; (2) the apparent clock, which maintains the time-of-day used by client processes and (3) the actual clock, which represents the time-of-day provided by an outside reference. The apparent and actual clocks are maintained as 48-bit quantities with 32 bits of significance available to client processes. These clocks run at a rate of 1000 Hz and are reset at midnight UT.

The CLOCK process consists of a set of state variables along with a set of procedures that are called as the result of hardware interrupts and client requests. An interval timer is assumed logically separate from the local clock mechanism, although both could be derived from the same timing source.

3.2.1. Local Variables

CLK.CLOCK
: This is a 48-bit fixed-point state variable used to represent the apparent time-of-day. The decimal point is to the right of bit 16 (numbering from the right at bit 0). Bit 16 increments at a rate equivalent to 1000 Hz independent of the hardware clock. (In the case of programmable-clock hardware the value of CLK.CLOCK must be corrected as described below.)

CLK.COUNT
: This is a hardware register that increments at rate R. It can be represented by a simple line clock, which causes interrupts at the line-frequency rate, or by a programmable clock, which contains a 16-bit register that is programmed to count at a 1000-Hz rate and causes an interrupt on overflow. The register is considered a fixed-point variable with decimal point to the right of bit 0.

CLK.DELTA
: This is a 48-bit signed fixed-point state variable used to represent the increment to be added to CLK.CLOCK to yield the actual time-of-day. The decimal point is to the right of bit 16.

3.2.3. Parameters

ADJUST-FRACTION
: This is an integer which defines the shift count used to compute a fraction that is used as a multiplier of CLK.DELTA to correct CLK.CLOCK once each clock-adjust interval. A value of seven is suggested.

ADJUST-INTERVAL
: This is an integer which defines the clock-adjust interval in milliseconds. A value of 500 (one-half second) is suggested for the line clock and 4000 (four seconds) for the 1000-Hz clock.

CLOCK-TICK
This is a fixed-point integer which defines the increment in milliseconds to be added to CLK.CLOCK as the result of a clock tick. The decimal point is to the right of bit 16. In the case of a line-clock interrupt, the value of CLOCK-TICK should be 16.66666 (60 Hz) or 20.00000 (50 Hz). In the case of a 1000-Hz programmable-clock overflow, the value should be 65536.00000.

HOLD-INTERVAL
This is an integer which defines the number of seconds that HOLD will count down after CLK.CLOCK has been reset. The resulting interval must be at least as long as the maximum HELLO-INTERVAL used by any HELLO process.

3.2.3. Events and Procedures

INCREMENT-CLOCK Event
This event is evoked as the result of a tick interrupt, in the case of a line clock, or a counter overflow, in the case of the 1000-Hz clock. It causes the logical clock to be incremented by the value of CLOCK-TICK.

1. Add the value of CLOCK-TICK to CLK.CLOCK.

ADJUST-CLOCK Event
This event is evoked once every ADJUST-INTERVAL milliseocnds to slew the apparent clock time to the actual clock time as set by the SET-CLOCK procedure. This is done by subtracting a fraction of the correction factor CLK.DELTA from the value of CLK.DELTA and adding the same fraction to CLK.CLOCK. This continues until either the next SET-CLOCK call or CLK.DELTA has been reduced to zero.

The suggested values for ADJUST-INTERVAL and ADJUST-FRACTION represent a maximum slew rate of less than +-2 milliseconds per second, in the case of the line clock, and +-250 microseconds per second, in the case of the 1000-Hz clock. The action is to smooth noisy clock corrections received from neighboring systems to obtain a high-quality local reference, while insuring the apparent clock time is always monotonically increasing.

1. Shift the 48-bit value of CLK.DELTA arithmetically ADJUST-FRACTION bits to the right, discarding bits from the right and saving the result in a temporary variable F. Assuming the decimal point of F to be positioned to the right of bit 16 and sign-extending as necessary, subtract F from CLK.DELTA and add F to CLK.CLOCK.

DECREMENT-HOLD Event
This event is evoked once per second to decrement the value of HOLD.

1. If the value of HOLD is zero, do nothing; otherwise, decrement its value.

READ-CLOCK Procedure

This procedure is called by a client process. It returns the apparent time-of-day computed as the integer part of the sum CLK.CLOCK plus CLK.COUNT. Note that the precision of the value returned is limited to +-1 millisecond, so that client processes must expect the apparent time to "run backward" occasionally due to drift correctins. When this happens the backward step will never be greater than one millisecond and will never occur more often than twice per second.

1. In the case of line clocks CLK.COUNT is always zero, while in the case of programmable clocks the hardware must be interrogated to extract the value of CLK.COUNT. If following interrogation a counter-overflow condition is evident, add CLOCK-TICK to CLK.CLOCK and interrogate the hardware again.

2. When the value of CLK.COUNT has been determined compute the sum CLK.COUNT + CLK.CLOCK. If this sum exceeds the number of milliseconds in 24 hours (86,400,000), reduce CLK.CLOCK by 86,400,000, set HOLD-INTERVAL -> HOLD, set CLOCK-VALID (bit 15 of DATE) to one, roll over DATE to the next calender day and start over. If not, return the integer part of the sum as the apparent time-of-day.

   The CLOCK-VALID bit is set to insure that a master-clock update is received at least once per day. Note that, in the case of uncompensated crystal oscillators of the type commonly used as the 1000-Hz time base, a drift of several parts per million can be expected, which would result in a time drift of several tenths of a second per day, if not corrected.

SET-CLOCK Procedure

This procedure is called by a client process. It sets a time-of-day correction factor in milliseconds. The argument represents a 32-bit signed fixed-point quantity with decimal point to the right of bit 0 that is to be added to CLK.CLOCK so that READ-CLOCK subsequently returns the actual time-of-day.

1. If the correction factor is in the range -2**(16-ADJUST-FRACTION) to +2**(16-ADJUST-FRACTION) - 1 (about +-128 milliseconds with the suggested value of ADJUST-FRACTION), the value of the argument replaces CLK.DELTA and the procedure is complete. If not, add the value of the sign-extended argument to CLK.CLOCK and set CLK.DELTA to zero. In addition, set HOLD-INTERVAL -> HOLD, since this represents a relatively large step-change in apparent time. The value of HOLD represents the remaining number of seconds in which timestamps should be considered invalid and is used by the HELLO process to suppress roundtrip delay calculations which might involve invalid timestamps.

3.3. HELLO Process

The HELLO process maintains clock synchronization with a neighbor HELLO process using the HELLO protocol. It also participates in the routing

algorithm. There is one HELLO process and one set of local state variables for each link connecting the host to one of its neighbors.

3.3.1. Local variables

HLO.BROADCAST
: This is a one-bit switch state variable. When set to zero a point-to-point link is assumed. When set ot one a broadcast (e.g. Ethernet) link is assumed.

HLO.KEEP-ALIVE
: This is an eight-bit counter state variable used to indicate whether the link is up. It is initialized with a value of zero.

HLO.LENGTH
: This is a 16-bit integer state variable used to record the length in octets of the last HELLO message sent.

HLO.NEIGHBOR-ADDRESS
: This is a 32-bit integer state variable used to contain the neighbor host Internet address.

HLO.PID
: This is an eight-bit integer state variable used to identify the net-output process associated with this HELLO process. It is initialized by the kernel when the process is created and remains unchanged thereafter.

HLO.POLL
: This is a one-bit switch state variable. When set the HELLO process spontaneously sends HELLO messages. When not set the HELLO process responds to HELLO messages, but does not send them spontaneously.

HLO.TIMESTAMP
: This is a 32-bit integer temporary variable used to record the time of arrival of a HELLO message.

HLO.TSP
: This is a 16-bit signed integer state variable used in roundtrip delay calculations.

3.3.2. Parameters

HELLO-INTERVAL
: This is an integer which defines the interval in seconds between HELLO messages. It ranges from about eight to a maximum of 30 seconds, depending on line speed.

HOLD-DOWN-INTERVAL
: This is an integer which defines the interval in seconds a host will be

considered up following receipt of a HELLO message indicating that host is up. A value of 120 is suggested.

KEEP-ALIVE-INTERVAL
This is an integer which defines the interval, in units of HELLO-INTERVAL, that a HELLO process will consider the link up. A value of four is suggested.

MAXDELAY
This is an integer which defines the maximum roundtrip delay in seconds on a path to any reachable host. A value of 30 is suggested.

MINDELAY
This is an integer which defines the minimum switching threshold in milliseconds below which routes will not be changed. A value of 100 is suggested.

3.3.3. Events and Procedures

INPUT-PACKET Event
When a packet arrives the net-input process first sets HLO.TIMESTAMP to the value returned by the READ-CLOCK procedure, then checks the packet for valid local leader, IP header format and checksum. If the protocol field in the IP header indicates a HELLO message, the packet is passed to the HELLO process. If any errors are found the packet is dropped.

The HELLO process first checks the packet for valid HELLO header format and checksum. If any errors are found the packet is dropped. Otherwise, it proceeds as follows:

1. If PKT.SOURCE is equal to LOCAL-ADDRESS, then the line to the neighbor host is looped. If this is a broadcast link (HLO.BROADCAST is set to one), then ignore this nicety; if not, this is considered an error and further processing is abandoned. Note that, in special configurations involving other systems (e.g. ARPANET IMPs and gateways) it may be useful to use looped HELLO to monitor connectivity. The DCN implementation provides this feature, but is not described here.

2. Set KEEP-ALIVE-INTERVAL -> HLO.KEEP-ALIVE. This indicates the maximum number of HELLO intervals the HLO.TSP field is considered valid.

3. Set PKT.TIMESTAMP - HLO.TIMESTAMP -> HLO.TSP. This is part of the roundtrip delay calculation. The value of HLO.TSP will be updated and returned to the neighbor in the next HELLO message transmitted. Next, compute the raw roundtrip delay and offset:
HLO.TIMESTAMP - PKT.TSP -> DELAY and HLO.TSP + DELAY/2 -> OFFSET.
Note: in the case of a broadcast link (HLO.BROADCAST set to one) set DELAY to zero.

4. Perform this step only in the case of non-broadcast links (HLO.BROADCAST set to zero). If PKT.SOURCE is not equal to HLO.NEIGHBOR-ADDRESS, then a new neighbor has appeared on this link. Set PKT.SOURCE -> HLO.NEIGHBOR ADDRESS, MAXDELAY -> DELAY and proceed to the next step. This will force the line to be declared down and result in a hold-down cycle. Otherwise, if either PKT.TSP is zero or HOLD is nonzero, then the DELAY calculation is invalid and further processing is abandoned. Note that a hold-down cycle is forced in any case if a new neigbor is recognized.

5. If processing reaches this point the DELAY and OFFSET variables can be assumed valid as well as the remaining data in the packet. First, if DELAY < MINDELAY, set MINDELAY -> DELAY. This avoids needless path switching when the difference in delays is not significant and has the effect that on low-delay links the routing algorithm degenerates to min-hop rather than min-delay. Then set HLO.PID -> PID. There are two cases:

   Case 1: PKT.NHOSTS is zero.
   This will be the case when the neighbor host has just come up or is on a different net or subnet. Set NEIGHBOR-ADDRESS -> ADDRESS and call the ROUTE procedure, which will return the host ID. Then call the UPDATE procedure. In the case of errors, do nothing but return.

   Case 2: PKT.NHOSTS is nonzero.
   This is the case when the neighbor host is on the same net or subnet. First, save the values of DELAY and OFFSET in temporary variables F and G. Then, for each value of HID from zero to NHOSTS-1 consider the corresponding PKT.HOSTS-TABLE entry and do the following: Set F + PKT.HOST-TABLE.DELAY -> DELAY and G + PKT.HOST-TABLE.OFFSET -> OFFSET and call the UPDATE procedure. This completes processing.

   ROUTE Procedure
   This procedure returns the host ID in HID of the host represented by the global variable ADDRESS.

1. First, determine if the host represented by ADDRESS is on the same local net as LOCAL-ADDRESS. For the purposes of this comparison bits 0-7 and 16-31 are compared for class-A nets and bits 8-31 are compared for class-B and class-C nets. This provides for a subnet capability, where the bits 0-7 and 16-23 (class-A) or 8-15 (class-B) are used as a subnet number.

   Case 1: The host is on the same net or subnet.
   Extract the address field of ADDRESS, subtract ADDRESS-OFFSET and store the result in HID. If 0 <= HID < NHOSTS, the procedure completes normally; otherwise it terminates in an error condition.

Case 2: The host is not on the same net or subnet.
Search the NET-TABLE for a match of the net fields of LOCAL-ADDRESS and NET-TABLE.NET. If found set NET-TABLE.HID -> HID and return normally. If the NET-TABLE.NET field is zero, indicating the last entry in the table, set HET-TABLE.HID -> HID and return normally. Note that, in the case of hosts including GGP/EGP gateway modules, if no match is found the procedure terminates in an error condition.

UPDATE Procedure

This procedure updates the entry of HOST-TABLE indicated by HID using three global variables: DELAY, OFFSET and PID. Its purpose is to update the HOST-TABLE entry corresponding to host ID HID. In the following all references are to this entry.

1. If PID is not equal to HOST-TABLE.PID, the route to host HID is not via the net-output process associated with this HELLO process. In this case, if DELAY + MINDELAY > HOST-TABLE.DELAY, the path is longer than one already in HOST-TABLE, so the procedure does nothing.

2. This step is reached only if either the route to host HID is via the net-output process associated with this HELLO process or the newly reported path to this host is shorter by at least MINDELAY. There are two cases:

   Case 1: HOST-TABLE.DELAY < MAXDELAY.
   The existing path to host HID is up and this is a point-to-point link (HLO.BROADCAST is set to zero). If DELAY < MAXDELAY the newly reported path is also up. Proceed to the next step. Otherwise, initiate a hold-down cycle by setting MAXDELAY -> HOST-TABLE.DELAY and HOLD-DOWN-INTERVAL -> HOST-TABLE.TTL and return.

   Case 2: HOST-TABLE.DELAY >= MAXDELAY.
   The existing path to host HID is down. If DELAY < MAXDELAY and HOST-TABLE.TTL is zero, the hold-down period has expired and the newly reported path has just come up. Proceed to the next step. Otherwise simply return.

3. In this step the HOST-DELAY entry is updated. Set DELAY -> HOST-TABLE.DELAY, HOLD-DOWN-INTERVAL -> HOST-TABLE.TTL and HLO.PID -> HOST-TABLE.PID.

4. For precise timekeeping, the offset can be considered valid only if the length of the last HELLO packet transmitted is equal to the length of the last one received. Thus, if HLO.LENGTH equal to PKT.LENGTH, set OFFSET -> HOST-TABLE.OFFSET; otherwise, leave this field alone. Finally, if HID is equal to CLOCK-HID and bit 15 (the DATE-VALID bit) of DATE is zero, set PKT.DATESTAMP -> DATE and call the SET-CLOCK procedure of the CLOCK process with argument HLO.TIMESTAMP.

OUTPUT-PACKET Event

This event is evoked once every HELLO-INTERVAL seconds. It determines if a HELLO message is to be transmitted, transmits it and updates state variables.

1. If HLO.KEEP-ALIVE is nonzero decrement its value.

2. If HLO.POLL is zero and HLO.KEEP-ALIVE is zero, do not send a HELLO message. If either is nonzero initialize the packet fields as follows: LOCAL-ADDRESS -> PKT.SOURCE, HLO.NEIGHBOR-ADDRESS -> PKT.DESTINATION and DATE -> PKT.DATESTAMP. Note: PKT.DESTINATION is set to zero if this is a broadcast link (HLO.BROADCAST set to one). Also, note that bit 15 of DATE is the DATE-VALID bit. If this bit is one the receiver will not update its master clock from the information in the transmitted packet. This is significant only if the sending host is on the least-delay path to the master clock. Set PKT.TIMESTAMP to the value returned from the READ-CLOCK procedure. If HLO.KEEP-ALIVE is zero or HOLD is nonzero, set PKT.TSP to zero; otherwise, set PKT.TIMESTAMP + HLO.TSP -> PKT.TSP.

3. Determine if the neighbor is on the same net or subnet. If the neighbor is on a different net set PKT.NHOSTS to zero and proceed with the next step. Otherwise, set NHOSTS -> PKT.NHOSTS and for each value of HID from zero to PKT.HOSTS-1 copy the HOST-TABLE.DELAY and HOST-TABLE.OFFSET fields of the corresponding HOST-TABLE entry in order into the packet. For each entry copied test if the HOST-TABLE.PID field matches the HLO.PID of the HELLO process. If so, a potential routing loop is possible. In this case use MAXDELAY for the delay field in the packet instead.

4. Finally, set HLO.LENGTH to the number of octets in the packet and send the packet.

3.4. HOST Process (HOS)

This process maintains the routing tables. It is activated once per second to scan HOST-TABLE and decrement the HOST-TABLE.TTL field of each entry. It also performs housekeeping functions.

3.4.1. Local variables

HOS.PID

This is an eight-bit integer used to identify the HOST process. It is initialized by the kernel when the process is created and remains unchanged thereafter.

HOS.HID

This is an eight-bit temporary variable.

3.4.2. Events and Procedures

SCAN Event

This event is evoked once each second to scan the HOST-TABLE and perform housekeeping functions.

1. For each value of a temporary variable F from zero to NHOSTS-1 do the following: Set LOCAL-ADDRESS -> ADDRESS and call the ROUTE procedure, which will return the host ID HID. If F is equal to HID, then set both DELAY and OFFSET to zero, HOS.PID -> PID and call the UPDATE procedure. This will cause all packets received with the local address to be routed to this process.

   If HOST-TABLE.TTL is zero skip this step. Otherwise, decrement HOST-TABLE.TTL by one. If the result is nonzero skip the remainder of this step. Otherwise, If HOST-TABLE.DELAY < MAXDELAY set HOLDOFF-INTERVAL -> HOST-TABLE.TTL and MAXDELAY -> HOST-TABLE.DELAY. The effect of this step is to declare a hold-down cycle when a host goes down.

4. References

1. Mills, D.L. Final Report on Internet Research, ARPA Packet Switching Program. Technical Report TSLAB 82-7, COMSAT Laboratories, December 1982.

Appendix A. Link-Level Packet Formats

A.1. Serial Links Using Program-Interrupt Interfaces

Following is a description of the frame format used on asynchronous and synchronous serial links with program-interrupt interfaces such as the DEC DLV11 and DPV11. This format provides transparency coding for all messages, including HELLO messages, but does not provide error detection or retransmission functions. It is designed to be easily implemented and compatible as far as possible with standard industry protocols.

The protocol is serial-by-bit, with the same interpretation on the order of transmission as standard asynchronous and synchronous interface devices; that is, the low-order bit of each octet is transmitted first. The data portion of the frame consists of one Internet datagram encoded according to a "character-stuffing" transparency convention:

1. The frame begins with the two-octet sequence DLE-STX, in the case of asynchronous links, or the four-octet sequence SYN-SYN-DLE-STX, in the case of synchronous links. The data portion is transmitted next, encoded as described below, followed by the two-octet sequence DLE-ETX. No checksum is transmitted or expected. If it is necessary for any reason to transmit time-fill other than in the data portion, the DEL (all ones) is used.

2. Within the data portion of the frame the transmit buffer is scanned for a DLE. Each DLE found causes the sequence DLE-DLE to be transmitted. If it is necessary for some reason for the transmitter to insert time-fill within the data portion, the sequence DLE-DEL is used.

3. While scanning the data stream within the data portion of the frame the sequence DLE-DLE is found, a single DLE is inserted in the receive buffer. If the sequence DLE-ETX is found, the buffer is passed on for processing. The sequence DLE-DEL is discarded. Any other two-octet sequence beginning with DLE and ending with other than DLE, ETX or DEL is considered a protocol error (see note below).

Note: In the case of synchronous links using program-interrupt interfaces such as the DPV11, for example, a slightly modified protocol is suggested when both ends of the link concur. These interfaces typically provide a parameter register which can be loaded with a code used both to detect the receiver synchronizing pattern and for time-fill when the transmit buffer register cannot be serviced in time for the next character.

The parameter register must be loaded with the SYN code for this protocol to work properly. However, *should* it be necessary to transmit time-fill, a single SYN will be transmitted, rather than the DLE-DEL sequence specified. Disruptions due to these events can be minimized by use of the following rules:

1. If the transmitter senses a time-fill condition (usually by a control bit assigned for this purpose) between frames or immediately following transmission of a DLE, the condition is ignored.

2. If the transmitter senses a time-fill condition at other times it sends the sequence DLE-CAN.

3. If the receiver finds a SYN either between frames or immediately followoing DLE, the SYN is discarded without affecting sequence decoding.

4. If the receiver finds the sequence DLE-CAN in the data portion, it discards the sequence and the immediately preceding octet.

These rules will work in cases where a single SYN has been inserted by the transmitter and even when a SYN has been inserted in the DLE-CAN sequence. If an overrun (lost data) condition is sensed at the receiver, the appropriate action is to return to the initial-synchronization state. This should also be the action if any code other than STX is found following the initial DLE. or if any code other than DLE, ETX, DEL or CAN is found following a DLE in the data portion.

A.2. Serial Links Using DDCMP Devices

Following is a description of the frame format used on DEC DDCMP links with DMA interfaces such as the DEC DMV11 and DMR11. These interfaces implement the DEC DDCMP protocol, which includes error detection and retransmission capabilities. The DDCMP frame format is as follows:

```
+-------------+-----+-----+-----+-----+-----+-----+------+------+
| SYN SYN SOH |Count|Flag |Resp | Seq | Adr | CRC1 | Data | CRC2 |
+-------------+-----+-----+-----+-----+-----+-----+------+------+
bits   24       14    2     8     8     8     16     ...    16
```

With respect to this diagram, each octet is transmitted starting from the leftmost octet, with the bits of each octet transmitted low-order bit first. The contents of all fields except the "Data" field are managed by the interface. The Internet datagram is placed in this field as-is, with no character or bit stuffing (the extent of this field is indicated by the interface in the "Count" field.

A.3. Serial Links Using HDLC Devices

Following is a description of the frame format used on HDLC links with program-interrupt interfaces such as the DEC DPV11.

```
        +--------+--------+--------+--------+--------+--------+
        |  Flag  |  Addr  |  Ctrl  |  Data  |  CRC   |  Flag  |
        +--------+--------+--------+--------+--------+--------+
coding   01111110 00000000 00000000 xxxxxxxx cccccccc 01111110
```

With respect to this diagram, each octet is transmitted starting from the leftmost octet, with the bits of each octet transmitted low-order bit first. The code xxxxxxxx represents the data portion and cccccccc represents the checksum. The bits between the "Flag" fields are encoded with a bit-stuffing convention in which a zero bit is stuffed following a string of five one bits. The "Addr" and "Ctrl" fields are not used and the checksum is ignored. The Internet datagram is placed in the "Data" field, which must be a multiple of eight bits in length.

A.4. ARPANET 1822 Links Using Local or Distant Host Interfaces

Following is a description of the frame format used with ARPANET 1822 Local or Distant Host interfaces. These interfaces can be used to connect a DCN host to an ARPANET IMP, Gateway or Port Expander or to connect two DCN hosts together. When used to connect a DCN host to an ARPANET IMP, Gateway or Port Expander, a 96-bit 1822 leader is prepended ahead of the Internet datagram. The coding of this leader is as described in BBN Report 1822. When used to connect two DCN hosts together, no leader is used and the frame contains only the Internet datagram.

A.5. ARPANET 1822 Links Using HDH Interfaces

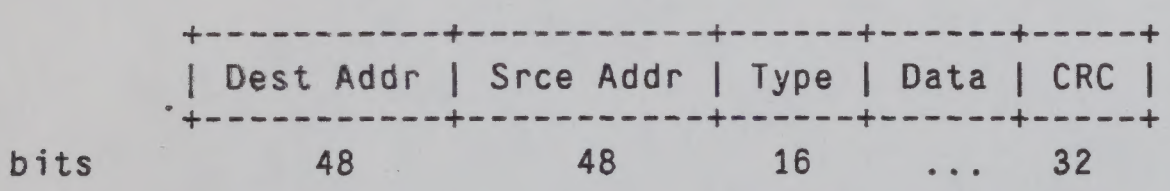
Following is a description of the frame format used with ARPANET 1822 HDH interfaces. These interfaces can be used to connect a DCN host to an ARPANET IMP or Gateway or to connect two DCN hosts together. In either case, the frame format is as described in Appendix J of BBN Report 1822.

A.6. X.25 LAPB Links Using RSRE Interfaces

Following is a description of the frame format used on X.25 LAPB links with the Royal Signals and Radar Establishment interfaces. These interfaces implement the X.25 Link Access Protocol - Balanced (LAPB), also known as the frame-level protocol, using a frame format similar to that described under A.3 above. Internet datagrams are placed in the data portion of I frames and encoded with the bit-stuffing procedure described in A.3. There is no packet-level format used with these interfaces.

A.7. Ethernet Links

Following is a description of the frame format used on Ethernet links.



With respect to this diagram, each field is transmitted starting from the leftmost field, with the bits of each field transmitted low-order bit first. The "Dest Addr" and "Srce Addr" contain 48-bit Ethernet addresses, while the "Type" field contains the assigned value for IP datagrams (0800 hex) or for

ARP datagrams (0806 hex). The Internet datagram is placed in the "Data" field and followed by the 32-bit checksum. The Address Resolution Protocol (ARP) is used to establish the mapping between Ethernet address and Internet addresses.